366-61
202185
p. 6

19940047726

N94-71481

Aircraft Design Optimization Using a Quasi-Procedural Method and Expert System

Ilan Kroo and Masami Takai
Department of Aeronautics and Astronautics, Stanford University
Stanford, California

Introduction

This paper describes a new program architecture for complex engineering design and illustrates its application to aircraft design optimization. This *quasi-procedural* method selects and executes the analysis subroutines for the calculation of objective function and constraints. Furthermore, it decides which variables need to be recomputed in response to the change of a design variable, permitting the objective and constraints to be recalculated efficiently. A rule-based expert system is also used to identify active constraints and suggest solutions to make the design feasible. The integrated optimizer, quasi-procedural program, and expert system are applied to the aerodynamic optimization of a swept wing and to the complete synthesis of a medium range commercial aircraft. The performance of the system is compared with that of conventional programs.

Quasi-Procedural Method

The quasi-procedural method (Ref. 1) is a form of nonprocedural program, consisting of a set of small compiled subroutines and an executive routine that keeps track of the subroutine and variable dependencies. In response to a request for the value of a certain variable, the executive program calls the relevant routines in the appropriate order. In the example shown in figure 1, the system traces the variable dependencies, through intermediate results of many subroutines, from the desired output to the input variables shown at the top of the figure. This real-time arrangement of the computational path offers improved flexibility and extensibility compared with conventional procedural methods. In addition, because the structure of the computations is known, the system performs consistency maintenance, recognizing, for example, that changes in the variable B have no effect on the desired output and therefore do not require recomputation. This improves the system efficiency and is particularly significant when the quasi-procedural analysis is combined with numerical optimization.

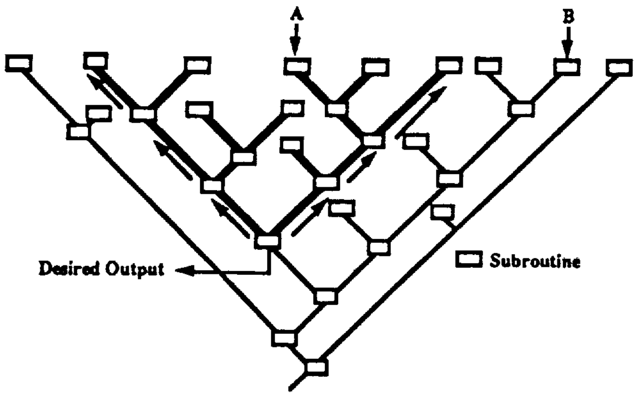

Figure 1. Quasi-Procedural Method and Consistency Maintenance

Numerical Optimization Using Quasi-Procedural Programming

Numerical optimization involves repeated evaluation of an objective function and constraints. At each evaluation, modification of some design variable (e.g., wing area) may require re-execution of a large number of analysis modules. On the other hand, changes in other variables, such as takeoff flap deflection, may invalidate only a few analyses. The quasi-procedural method recognizes the difference and reconfigures the computational path, allowing the optimizer to avoid the redundant calculations made by conventional methods with rigid program structures. This improved efficiency may be utilized in several ways. The following sections describe three methods by which the quasi-procedural method may be used to improve the performance of conventional numerical optimization methods.

Gradient Calculation

Time savings are especially large when the gradient of the objective function must be computed by finite differences. In this case, each component of the gradient is constructed by evaluating the change in objective function due to a change in the corresponding design variable. Since this process involves changing only a single design variable at a time, much of the computational path is unaffected and so the number of required computations is reduced. Furthermore, if the calculation of one gradient component requires a time T, which is substantially longer than the times required to compute the other components in a N-dimensional optimization, a fixed program structure evaluates the gradient in a time N*T, while the quasi-procedural method demands only a bit longer than T.

Constrained Optimization

Constrained optimization problems provide additional opportunities for reduction in computation time. Figure 2 shows a feasible region bounded by five constraints. The gradients of the objective and constraint functions are evaluated at the end of each line search. The size of a finite difference interval for gradient calculation must be small enough to approximate the derivatives accurately. This means that if a constraint is inactive at a point, that constraint is not violated during the gradient calculation at that point. The quasi-procedural method makes it easy to remove inactive constraints from gradient calculations. The resulting computational time savings is large when expensive constraints are inactive during much of the search. Further savings are achieved because not all design variables affect all constraints. Since changing take-off flap deflection affects climb and take-off field length, but not range or landing field length, no additional time is spent computing range when take-off flap is varied. This makes it possible to achieve the efficiency available with a reduced design variable set without actually changing design variables.

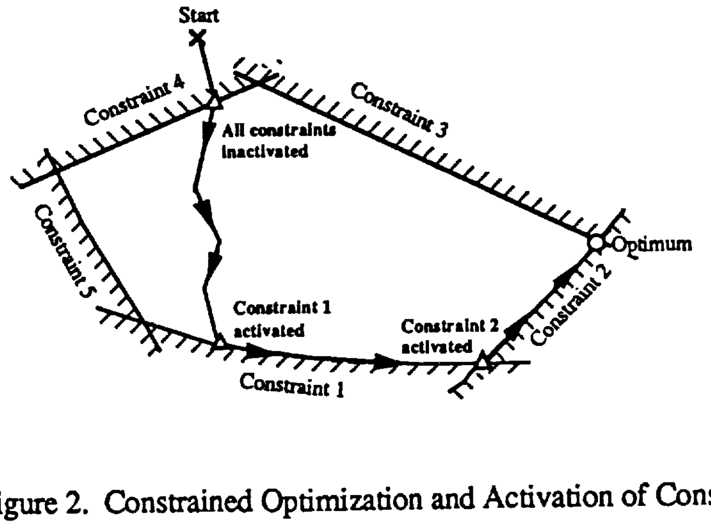

Figure 2. Constrained Optimization and Activation of Constriants

Use of the Chain Rule

The availability of subroutine dependency information, provided by the quasi-procedural method, enables the use of the chain rule, which may reduce the cost of derivative calculations (Ref. 2). Consider the analysis procedure shown in Figure 3. If the entire calculation were treated as a black box the time required for computation of the gradient of q with respect to the seven design variables would be: $7*(A+B+C+D+E+F+G+H+I)$ where A, B, C, etc. represent the time required for the corresponding routine (or twice this value if central-differencing is used.) However, given the dependency information, the derivative of q with respect to a may be written:

$$\begin{aligned}\partial q/\partial a &= \partial q/\partial o*\partial o/\partial a + \partial q/\partial p*\partial p/\partial a\\ &= \partial q/\partial o*(\partial o/\partial l*\partial l/\partial a + \partial o/\partial m*\partial m/\partial a + \partial o/\partial n*\partial n/\partial a)\\ &= \partial q/\partial o*(\partial o/\partial l*\partial l/\partial h*\partial h/\partial a + \partial o/\partial m*\partial m/\partial i*\partial i/\partial a)\end{aligned}$$

This evaluation is performed in the time: $I+H+E+A+H+F$.
The entire gradient requires a time of:
$(I+H+E+A+H+F) + A + (B+G+H) + B + (C+G) + (D+I) + F$
$= 2A+2B+C+D+E+2F+2G+3H+2I$
Thus the evaluation time is reduced from 63 to 16 (in subroutine units). The savings become more significant when one of the subroutines which is called only once (i.e., C, D, or E) is very time-consuming. For example, if the routine E requires 10 times the computational effort as the other routines, the calculation time is reduced by 80%.

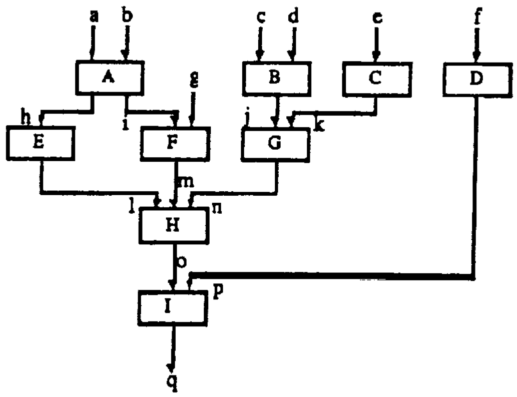

Figure 3. Gradient Calculation by Chain Rule and Intermediate Sensitivities

Expert System

Domain-specific knowledge can be especially useful in the early stages of aircraft design. Rather than start the optimization at an arbitrary point, an expert system is used to improve the initial design. This rule-based system was combined with the quasi-procedural program to warn the user of active constraints or other design problems, and to offer intelligent advice on how the problem might be corrected (Ref. 3).

The warning rules examine the current database, identify active design constraints, and report them to the user. A typical warning rule may compare the current value of a variable with its required value, and issue a warning string in case of constraint violation.

Solution rules analyze the causes of a constraint violation and generate solutions using design knowledge. A solution rule first looks at the warnings posted by the warning rules, and tries to identify a specific problem for which the solution rule is responsible. The rule may then look at the database to collect more information pertinent to the current problem and prescribe a solution best-suited for the current case. For example, a set of solution rules for the takeoff distance problem might be:

```
IF (TOFieldLength is too long) and not(ClimbGrad is too small)
and (*< TOFlapDefl 20.) THEN (Increase TOFlapDefl)

IF  (TOFieldLength is too long) and (*/ TotalSLST MaxTOW $TroverW)
and (*< $ToverW 0.2) THEN (Increase SLSThrust)
```

A rule-base with approximately 100 rules was used to resolve fundamental problems with the initial design so that the numerical optimization could be started in a feasible region.

Applications

The quasi-procedural analysis method was combined with a variable-metric optimizer to illustrate the efficiency of the system in realistic design applications. In this section, two example problems are discussed: the design of a swept wing using linear potential theory, and an aircraft synthesis and sizing problem in which direct operating cost is minimized.

Wing Design

In this example, the linearity of an aerodynamic analysis routine is exploited by the quasi-procedural method and nonlinear optimizer. The wing twist distribution is to be designed so that the induced drag is kept low, the lift coefficient distribution is relatively uniform, and the desired wing lift is achieved. The design variables include the wing taper ratio, the twist angle of each of the twenty panels, and the angle of attack. A quasi-Newton optimizer (Ref. 4) was used to minimize the objective function with a central differencing scheme for gradient calculation.

Figure 4 shows the geometry and vortex arrangement of the swept wing. The spanwise lift distribution and induced drag are computed based on a Weissinger method, using a discrete vortex representation of the wake and a concentrated bound vortex. The trailing vortices are evenly spaced along the span, and the discrete bound vortices are placed at the quarter chord. Using the Biot-Savart law, an aerodynamic influence coefficient (AIC) matrix is computed which relates the strengths of the discrete vortices Γ to the downwash w at the control points located at the three quarter chord of the spanwise panels: $AIC(i,j) = w_i$ due to unit vortex strength at j

The strengths of the bound vortices, and eventually the spanwise lift and C_l distributions, are then found by solving the linear equation:

$[AIC]\{\Gamma\} = U_\infty\{\theta\}$ where U_∞ is the freestream velocity.

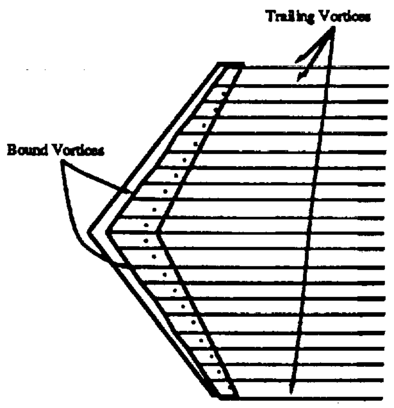

Figure 4. Wing Vortex Model

Figure 5 shows the program structure for the wing design problem. The subroutine `AIC` constructs the elements of the AIC matrix. `Decomp` performs LU decomposition of the AIC matrix, and `Solve` performs back-substitution. The AIC matrix has to be recomputed when taper ratio is modified, but need not be recalculated if only the twist angles and angle of attack are changed. The number shown to the right of each subroutine box in figure 5 is the execution time of the subroutine as a fraction of the total execution time. `Solve` is much faster than `AIC`, accounting for only 16% of the total computational time. This computational structure permits the quasi-procedural method to efficiently compute the gradient components with respect to the 20 twist angles and angle of attack. Figure 6 shows the optimal C_l and lift distributions. Note that the optimal C_l distribution is nearly constant as desired, and the lift distribution is nearly elliptic as reflected in the span efficiency of 0.98.

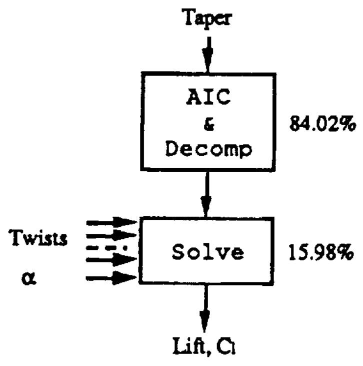

Figure 5. Structure of the Analysis Routines

Because a central differencing scheme is used, every gradient calculation requires evaluation of the AIC matrix twice each change in taper ratio, and solution of the linear system 42 times (2 times for taper ratio, and 40 times for twists). Therefore, the computational time for one gradient evaluation is:
2 * 5.33 sec + 42 * 1.01 sec = 53.2 sec. This figure may be compared with the time needed by conventional methods which do not recognize the structure of the program: 42 * (5.33 sec + 1.01 sec) = 266.3 sec. The ratio of these two values indicates a 80% saving by the quasi-procedural method.

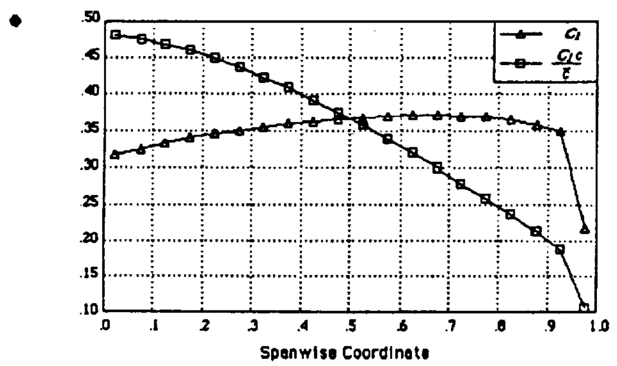

Figure 6. Optimal Lift and C_l Distributions

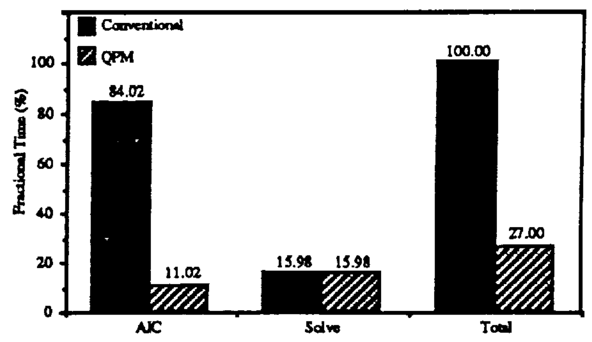

Figure 7. Computation Time Distribution

Figure 7 compares the total optimization times between the quasi-procedural and conventional methods. The figure shows the significant time saving obtained in the AIC calculation. The total computational saving is 73%, which is only 7% less than the maximum 80% gain available in the gradient calculation.

Complete Aircraft Synthesis

To illustrate the application of the method in a more complex problem, a set of aircraft analysis routines were written and used in the optimization of a medium range commercial aircraft. Ten design variables, shown in figure 8, were used to minimize the direct operating cost subject to constraints on range, landing and take-off field lengths, engine-out climb gradient, and cruise thrust. Figure 9 illustrates the major subroutines required for the calculation and their relationships with the objective and constraint routines. The analysis routines include geometric, aerodynamic, structural, and economic computations based on preliminary design methods of Douglas Aircraft Company (Ref. 5) which were modified for this application.

The expert system was first employed to generate a reasonable starting point for the numerical optimization. In this case, the system was able to suggest solutions to provide a feasible starting point. The variable metric optimizer was then used to minimize direct operating cost. Figure 8 shows the converged solution which satisfies the five constraints and reduces DOC by 4% compared with the initial design.

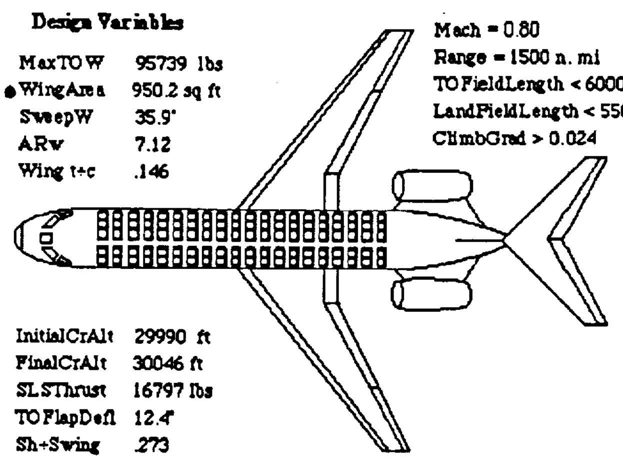

Figure 8. Optimal Geometry and Design Variables

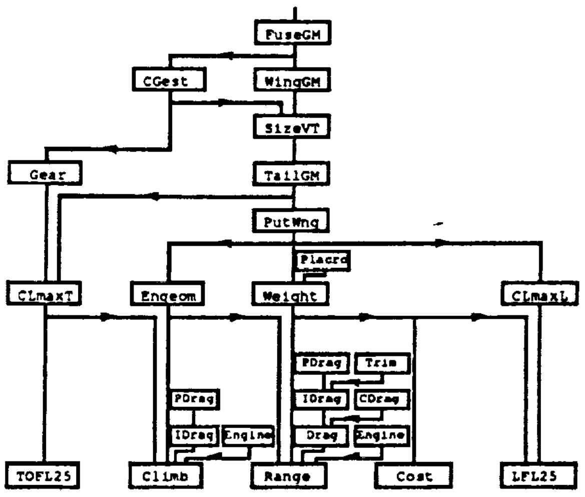

Figure 9. Computational Path for DOC and Constraints

Figure 10 shows the amount of time spent in each of the subroutines for three optimization cases. In the first case, the subroutines were arranged in a suitable order and the quasi-procedural system was not employed. In the second case, the system with consistency maintenance was used. Finally, inactive constraints were removed from gradient calculations as described previously. The result is an overall reduction in computation time of 22% for the quasi-procedural method, increasing to 39% when inactive constraints are removed. The figure shows that routines such as fuselage geometry (FUSEGM) are not affected by the selected design variables and so need to be executed only once.

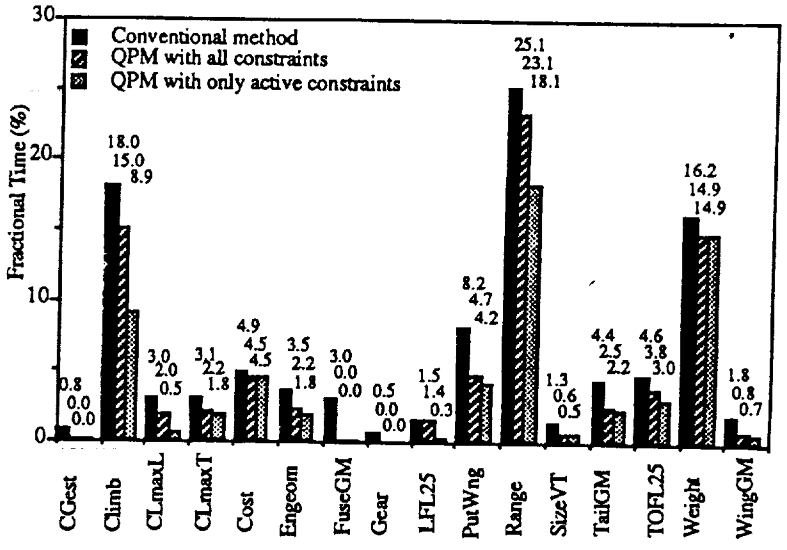

Figure 10. Distribution of Computation Times Among Analysis Routines

Conclusions and Continuing Work

The quasi-procedural method can significantly improve the performance of conventional numerical optimization. This is achieved primarily by the automatic simplification of the computational path during gradient calculations. In certain cases, savings of up to 80% in computation time are realized. Little improvement is seen during the line search portion of the optimization, however, since all variables are changed simultaneously at each step. Current work includes the investigation of non-gradient based methods (e.g. genetic algorithms) that may also be well-suited for use with quasi-procedural analysis.

References

1. Kroo, I., Takai, M., *"A Quasi-Procedural, Knowledge-Based System for Aircraft Design,"* AIAA Aircraft Design, Systems and Operations Meeting, AIAA paper No. 88-4428, September 1988.

2. Sobieski, J. S., *"Sensitivity of Complex, Internally Coupled Systems,"* AIAA Journal, Vol. 28, No. 1, p.153-160, January 1990.

3. Takai, M., A New Architecture and Expert System for Aircraft Design Synthesis, Stanford University, Ph.D. Thesis, June 1990.

4. Gill, P. E., Murray, W., Wright, M. H., Practical Optimization, Academic Press, London, 1981.

5. Kroo, I., *"Tail Sizing for Fuel-Efficient Transports,"* AIAA paper No. 83-2476, October 1983.